CSC

ICT Solutions for Brilliant Minds

GPU Programming. When, Why and How?

2024

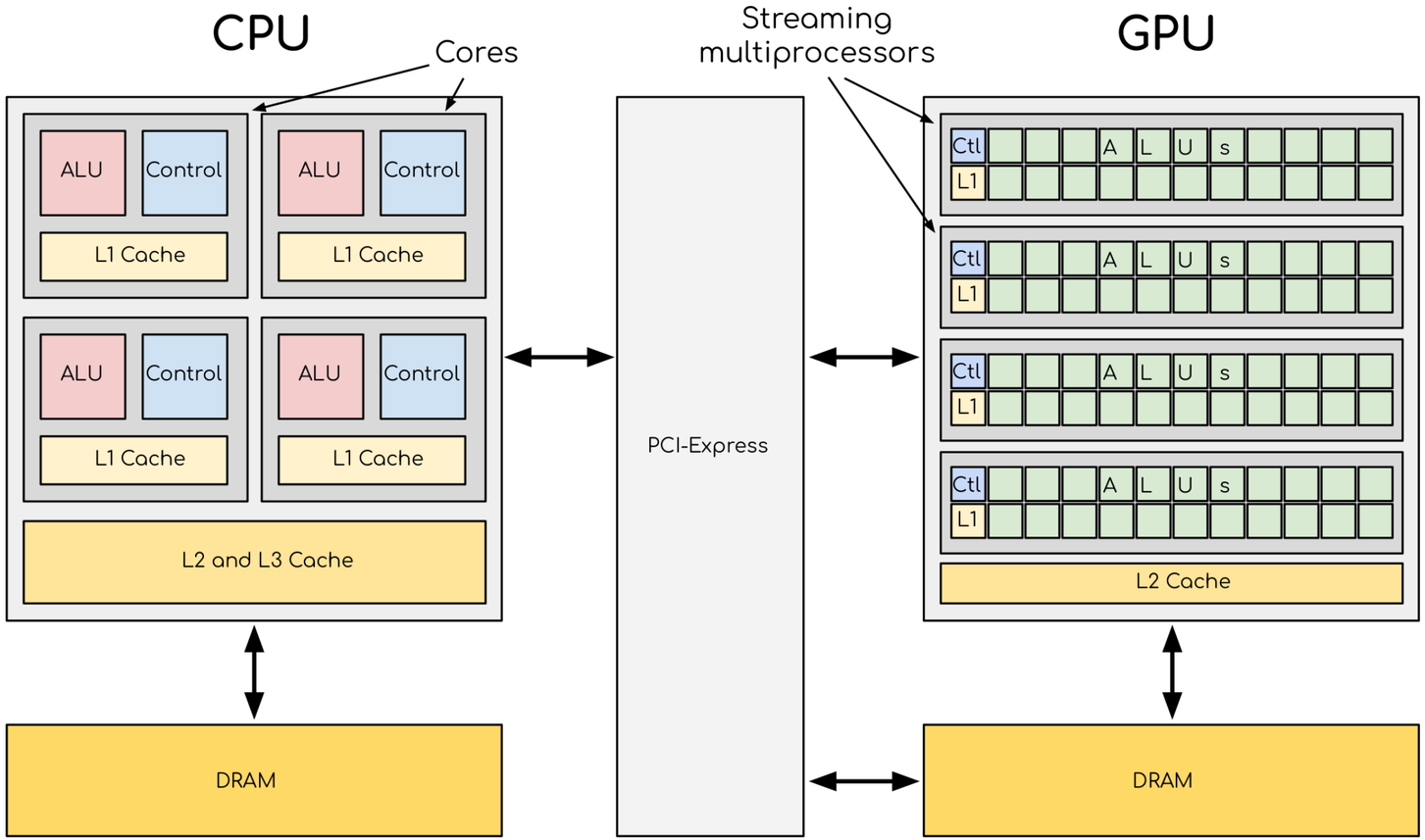
ENCCS Training

The GPU hardware and software ecosystem

Why GPUs?

- Speed: GPU computing can significantly accelerate many types of scientific workloads.
- Improved energy efficiency: GPUs can perform more calculations per watt of power consumed
- Cost-effectiveness: for certain workloads GPUs can be more cost-effective than traditional CPU-based systems.

Overview of GPU hardware



CPU (left) has complex core structure and pack several cores on a single chip. GPU cores are very simple in comparison, they also share data and control between each other.

Different design philosophies: CPU

- General purpose
- Good for serial processing
- Great for task parallelism
- Low latency per thread
- Large area dedicated cache and control
- Good for control-flow

Different design philosophies: GPU

- Highly specialized for parallelism
- Good for parallel processing
- Great for data parallelism
- High-throughput
- Hundreds of floating-point execution units
- Bad at control-flow processing

GPU Platforms

- *NVIDIA*, *AMD*, and *Intel* are the major conmpanies which design anf produce GPUs for HPC.
- GPUs are provided with their own suite **CUDA**, **ROCm**, and respectively **oneAPI**.
  - optimization, differentiation (offering unique features tailored to their devices), vendor lock-in, licensing, and royalty fees
- Cross-platform APIs such DirectCompute (only for Windows operating system), OpenCL, and SYCL.

Compute Unified Device Architecture (CUDA)

- NVIDIA's parallel computing platform.
  - Components: CUDA Toolkit & CUDA driver.
- Compilers: **nvcc**, **nvc**, **nvc++**, **nvfortran**.
  - Support GPU and multicore CPU programming (using OpenACC and OpenMP).
- CUDA API Libraries: `cuBLAS`, `cuFFT`, `cuRAND`, `cuSPARSE`.
- Debugging tools: **cuda-gdb**, **compute-sanitizer**.
- Performance analysis tools: **NVIDIA Nsight Systems**, **NVIDIA Nsight Compute**.
- Comprehensive CUDA ecosystem with extensive tools and features.

ROCm

- Open software platform for AMD accelerators.
  - Offers libraries, compilers, and development tools for AMD GPUs.
  - Supports C, C++, and Fortran languages.
  - Supports GPU and multicore CPU programming (using OpenMP and OpenCL).
- Debugging: **roc-gdb** command line tool.
- Performance analysis: **rocprof** and **roctracer** tools.
- **Heterogeneous-Computing Interface for Portability** (HIP).
  - Enables source portability for NVIDIA and AMD platforms, Intel in plan.
- Libraries: Prefixed with `roc` for AMD platforms
  - `hip`-prefixed wrappers ensure portability with no performance cost.

OneAPI

- Unified software toolkit for optimizing and deploying applications across various architectures, including CPUs, GPUs, and FPGAs.
  - Big focus on code reusability, performance, and portability.
- oneAPI **Base Toolkit**: Core set of tools and libraries for high-performance, data-centric applications. Includes SYCL support.
- oneAPI **HPC Toolkit**: additional compilers, debugging tools, MPI library, and performance analysis tool.
- Multiple programming models and languages supported: OpenMP, Classic Fortran, C++, SYCL.
- The code is portable to other OpenMP and SYCL frameworks.

Differences and similarities

- GPUs support different features, even among the same producer.
  - newer cards come with extra features and old features can become obsolete.
- Binaries have to be created targeting the specific architecture.
- **CUDA**: `-arch=sm_XY`
- **HIP** on *Nvidia*: `--gpu-architecture=sm_XY`
- **HIP** on *AMD*: `--offload-arch=gfxabc`

Terminology

NVIDIA	AMD	Intel
Streaming processor/streaming core	SIMD lane	processing element
SIMT unit	SIMD unit	Vector engine (XVE)
Streaming Multiprocessor (SMP)	Computing Unit (CU)	Xe-core / Execution unit (EU)
GPU processing clusters (GPC)	Compute Engine	Xe-slice

Summary

- GPUs are designed to execute thousands of threads simultaneously, making them highly parallel processors. In contrast, CPUs excel at executing a smaller number of threads in parallel.
- GPUs allocate a larger portion of transistors to data processing rather than data caching and flow control. This prioritization of data processing enables GPUs to effectively handle parallel computations and hide memory access latencies through computation.
- GPU producers provide comprehensive toolkits, libraries, and compilers for developing high-performance applications that leverage the parallel processing power of GPUs. Examples include CUDA (NVIDIA), ROCm (AMD), and oneAPI (Intel).
- These platforms offer debugging tools (e.g., `cuda-gdb`, `rocgdb`) and performance analysis tools (e.g., NVIDIA Nsight Systems, NVIDIA Nsight Compute, `rocprof`, `roctracer`) to facilitate code optimization and ensure efficient utilization of GPU resources.